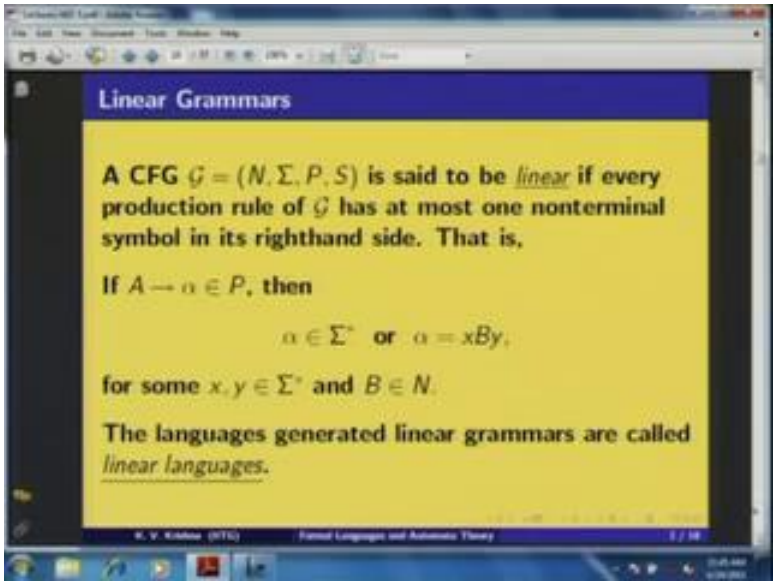
Formal Languages and Automata Theory
Prof. Dr. K. V. Krishna
Department of Mathematics
Indian Institute of Technology, Guwahati

Module - 02
Grammars
Lecture - 03
Regular Grammars

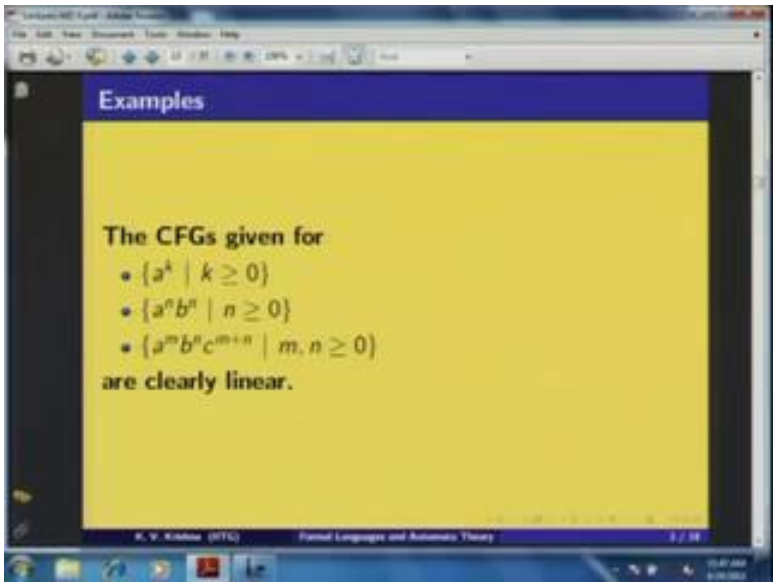
We are going to introduce an important special class of contextual grammars called linear grammar. Further we discuss even restricted class, which are useful to understand regular languages better.

(Refer Slide Time: 00:46)



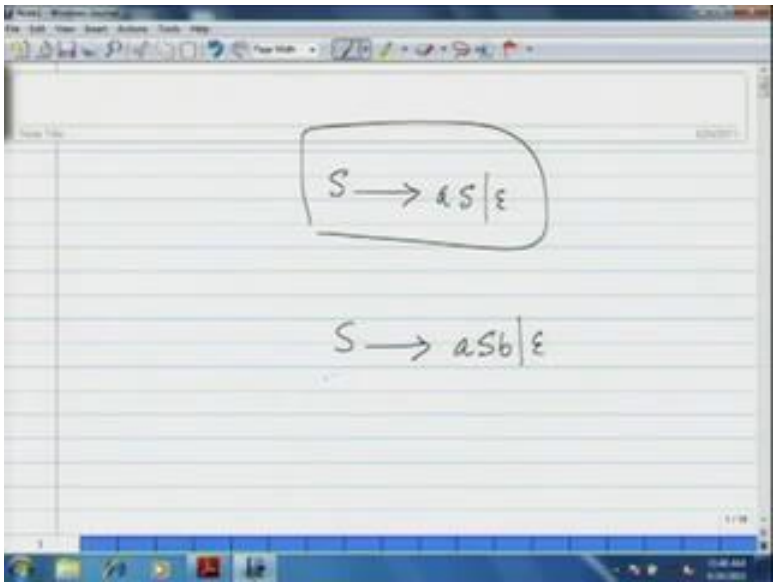
What are linear grammars? A gram a CFG N, sigma, P, S is said to be linear, if every production rule of g has at most one non terminal symbol in its right hand side. That is, if you take any production rule, A goes to alpha then either alpha is a terminal string or it is of the form x B y for some terminal strings x, y and B a non terminal symbol. Now, accordingly we call the languages generated by linear grammars are linear languages.

(Refer Slide Time: 01:28)



Let us look at some examples, we have CFG's for a power k k greater than or equal to 0.

(Refer Slide Time: 01:41)

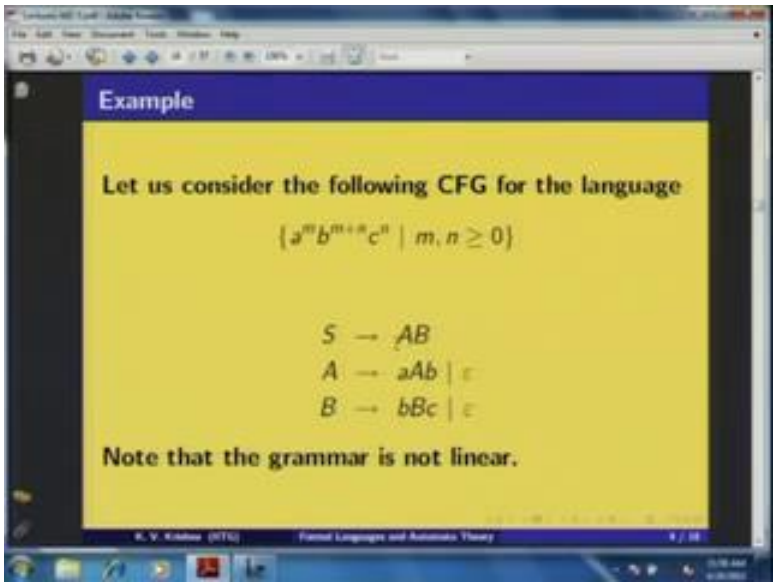


What is the grammar that we have introduced for this language, we have discuss like this, this grammar generates a power k k greater than or equal to 0; and you can clearly see that there is at most one ordinal symbol on the right hand side of each production rule. Because, we said that either it is a terminal string or on the right hand side of production rule, there can be one non terminal symbol, so here there are two production rules, one is having empty string on its right hand side and other is having a s.

So, each production rule has at most one non terminal symbol on right hand side and this grammar, this CFG that generates the language a power k k greater than equal to 0 is a linear grammar. And we have introduced a grammar, contextual grammar for a power n, b power n and where the production rules we gave it like this, again you see that there is at most one non terminal symbol on the right hand of each product, there are two production rules here.

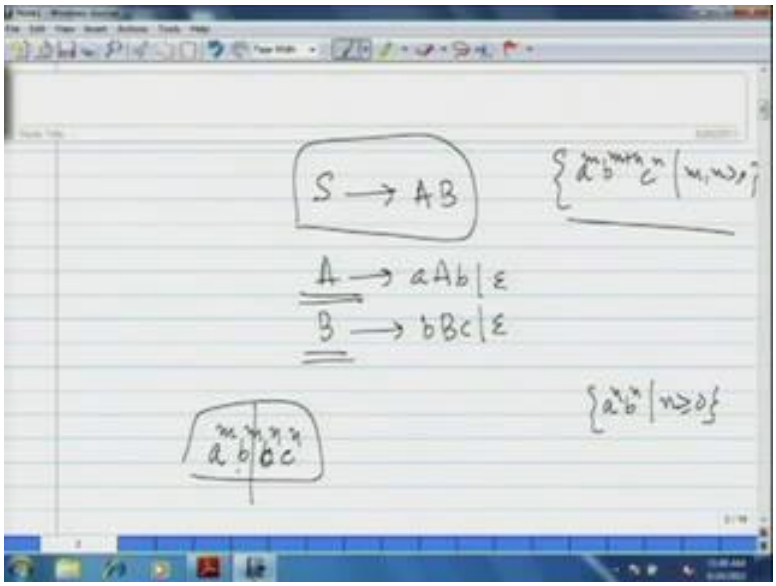
And thus you can see the grammar what we have introduced for the language a power n b power n greater than or equal to 0, for this that is also a linear grammar and hence, this language is linear. Also we have introduced a grammar for a power m b power m c power m plus n. for all m n positive, I mean non negative; so for this grammar what we have introduced you can clearly recollect and observe that clearly it is a linear grammar.

(Refer Slide Time: 03:34)



In contrast let us consider the following CFG for the language, because I ask you to work for a contextual grammar for this language. For example, one can consider suppose if I consider this contextual grammar for this language you can clearly see that S is S can A B this production rules.

(Refer Slide Time: 04:08)

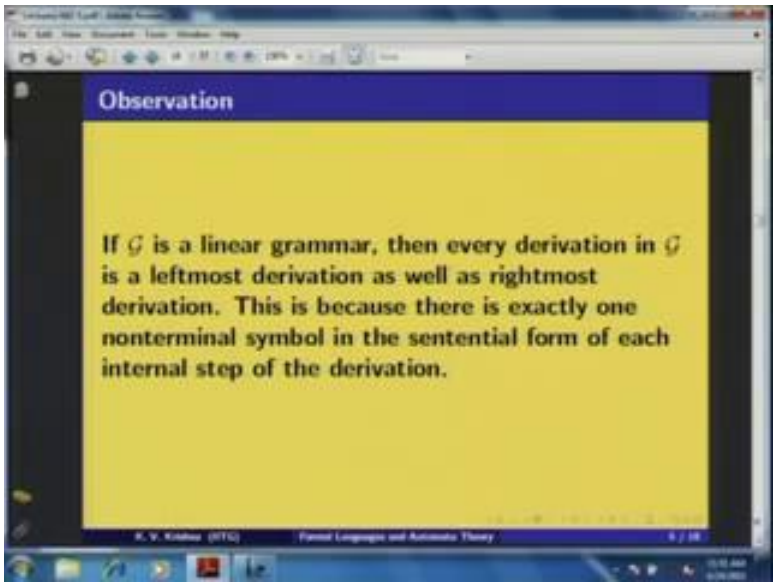


This S can be A B this production rule is having two portions A and B, and the portion A produces the string of form a power n b power n and b produces b power m c power m. So, that the desired string can be derived through this grammar, for example this B goes to b B c this how it is given or you can terminate. So, this actually following the philosophy behind the grammar with generates a power n b power n such that, m greater than or equal to 0 that example.

Only thing here, since we have to produce strings of the form a power m b power m plus n c power n, so any string here you can see a power m b power m, b power n c power n, that is what is the form. So, here you have two parts a power m b power m through the non terminal symbol A you can generate, and b power n c power n through the non terminal symbol b that you can generate. And if you consider the production rule S goes to A B this takes the concatenation of these two, and thus grammar can generate the language a power m b power m plus n c power n m n greater than or equal to 0.

And if you consider this grammar for this language, you see ((Refer Time: 06:09)) here is a production rule S goes to A B, on the right hand you have non terminal symbols and thus this grammar for this language is not linear. Because, every production rule should at most one non terminal symbol on it is right hand side, so here is a production rule in which you have two non terminal symbol on it is right hand side thus this grammar is not linear grammar.

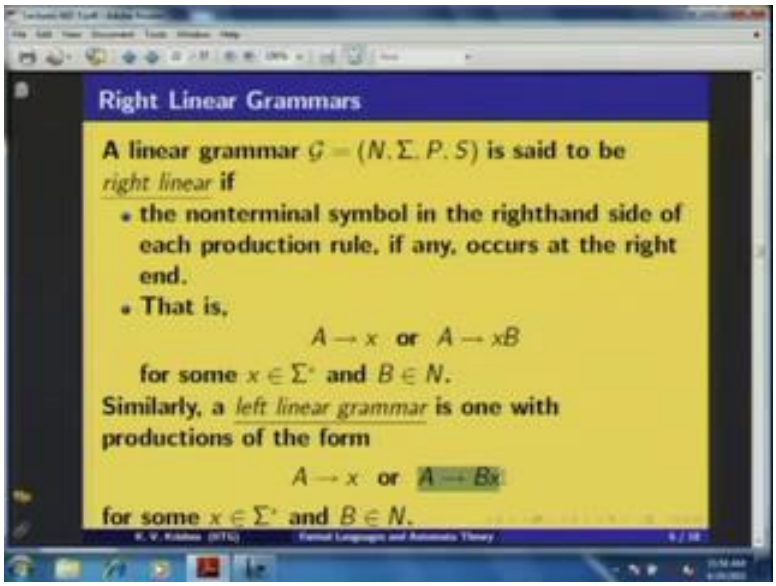
(Refer Slide Time: 06:36)



Now, you can observe this point, if g is a linear grammar, then every derivation in g is a left most derivation as well as right most derivation. The reason why here there is exactly one non terminal symbol in the sentential form of each internal step in the derivation, because every production rule has at most one non terminal symbol on its right hand side. So, each sentential form you will have only one non terminal symbol, thus when you apply further production rule you will be applying on it.

And thus what you can understand that is only non terminal symbol available on it is ((Refer Time: 07:20)) only one non terminal symbol will available in each sentential form of each step in a derivation. And thus you can treat it as first non terminal symbol that is left most non terminal symbol, that is as well as it is right most non terminal symbol. And thus each derivation is you can understand that it is a left most derivation as well as right most derivation.

(Refer Slide Time: 07:43)



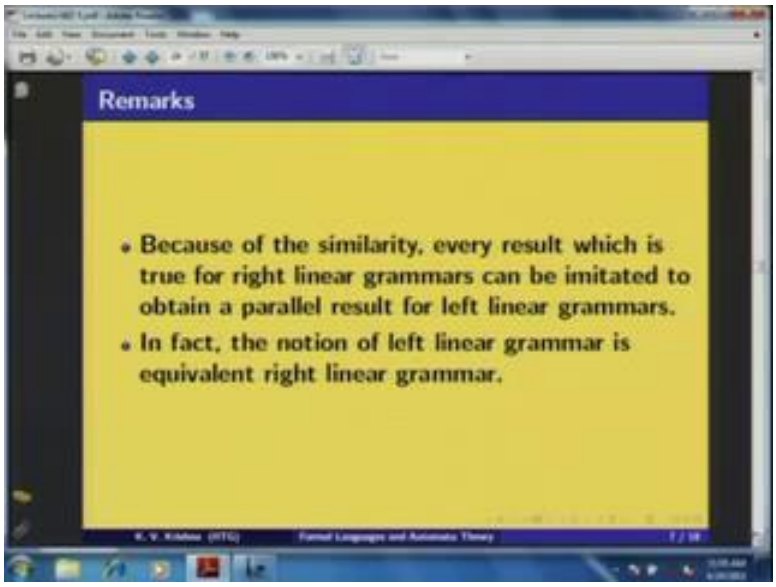
Now, we will introduce some more restrictions on this linear grammar to go further in the direction of regular grammars, and this regular grammars what we are going to introduce is essentially to give the facility to understand regular languages better. Because, in the beginning we have already mentioned that, the grammars that we are introducing here, in the context of regular languages to understand them better we are introducing, that is how we have mentioned.

Now, the tool CFG how languages can be generated and all those we have introduced and certain examples that I hope now you are familiar with; now the notion right linear grammar will give you the concept of regular grammars to understand regular languages better. So now how do we define this right linear grammar, a linear grammar you consider linear grammar that means, each production rule on it is right hand side you will have at most one non terminal symbol.

But, we put further restriction that is said to be right linear g said to be right linear, if the non terminal symbol on the right hand side of each production rule, if it is available it should occur on at the right end, that is the right hand side of each production rule is a terminal string followed by at most one non terminal symbol. That means, either it is of the form A goes to x, for x a terminal string or it is of the form x B, where x is a terminal string and B a non terminal symbol.

That means, a terminal string followed by at most one non terminal symbol, that is the restriction that we are considering and such linear grammar is called right linear grammar. Similarly, one can define left linear grammars in which each production rule, either right hand side of each production rule is of the form at most one non terminal symbol followed by a terminal string, so that means, A goes x or A goes B x of this form.

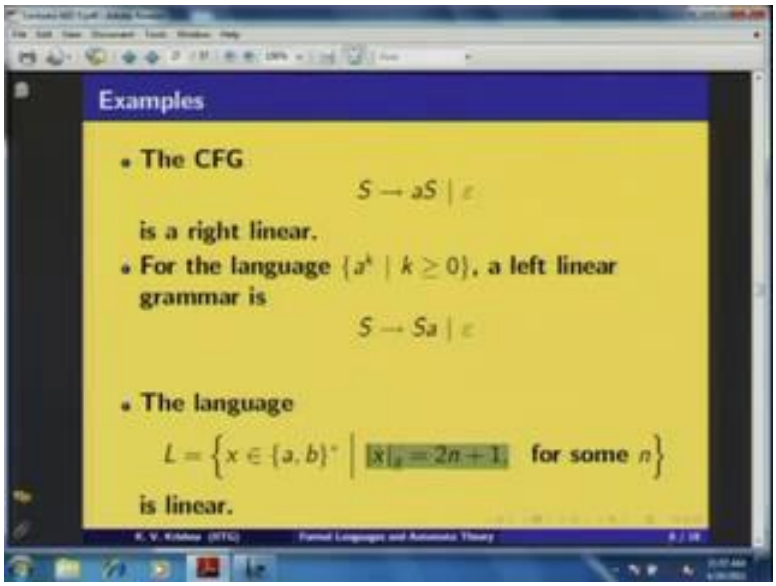
(Refer Slide Time: 10:04)



And now again you can see the similarity between these two definitions and thus any result that are I will observe for right linear grammars, one can imitate to get a parallel result for left linear grammars also. And in fact, we can understand, what we can observe is the notion of left linear grammars is equivalent to right linear grammars, what is the meaning of equivalence here, whatever is the language that you generate through a left linear grammar corresponding to that language you can construct a right linear grammar and conversely.

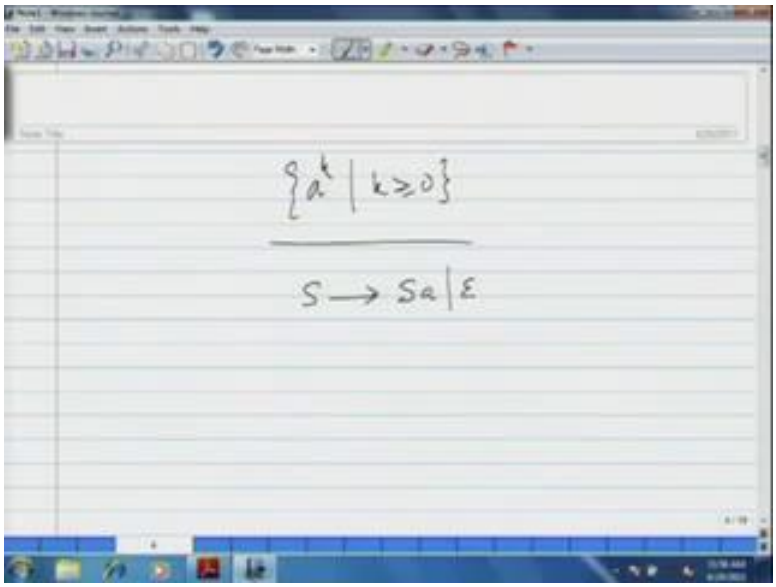
That means, for a right linear grammar whatever is the language that you have, corresponding to that language you can construct a left liner grammar, that is the equivalence here left linear grammars and right linear grammars are equivalent. To prove this result, we will be introducing some more tools and at that time I will recollect and we will remark once again; and how to understand that these two are equivalent, we will post pone the proof for the time being.

(Refer Slide Time: 11:09)



Now, let us look at some examples of right linear grammars, if you consider this contextual grammar S goes to a S or epsilon, what is the language generated by this grammar we have discussed this for several times.

(Refer Slide Time: 11:33)



That is the language a power k such that, k greater than or equal to 0, this is the language generated by ((Refer Time: 11:40)) this grammar. And you can clearly see that, any non terminal symbol that is appearing on right hand side of each production rule, that is appearing on a right most position of the production rule. And thus you can see by

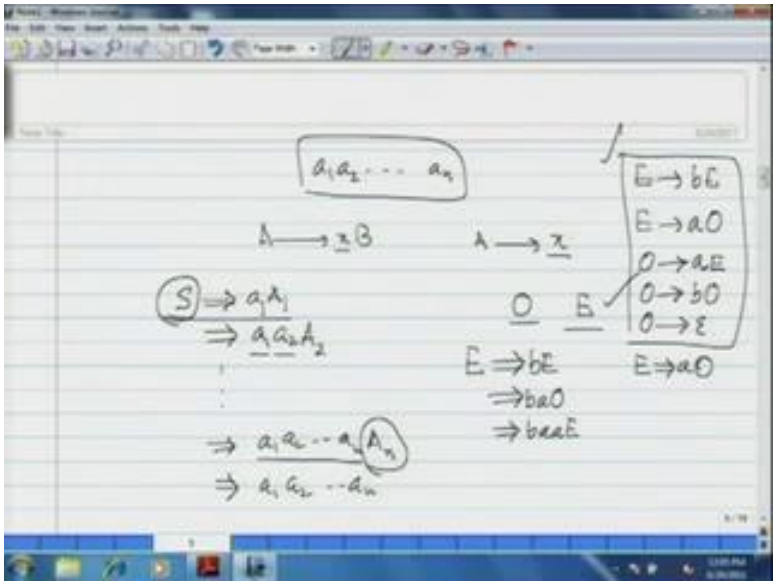
definition this is a right linear grammar and the language generated by this is a power k such that k greater than or equal to 0.

Now, as I have remarked that left linear grammars and right linear grammars are equivalent, at least for this example now I will give a left linear grammar also, here to understand that you have a left linear grammar for this language. If you consider the production rule, which is of the form S goes to S a or epsilon you can quickly understand that you can generate this language, because the production rule this time what I am considering is s a or epsilon.

And what are the way that I have proved that this grammar, this grammar generates a power k k greater than or equal to 0, you can imitate the same and understand, that this grammar also generates the same language and this is a left linear grammar. Now, let us consider this language set of all those strings over a, b with odd number of a is, because number of a is in x is of the form two n plus 1, that means there are odd number of a is.

I am claiming that it is a linear language, in fact you can have a right linear grammar for this of course, equivalently we have a left linear grammar for this language. For example, by know you understood that what is the importance of non terminal symbols in the production rules, like how we are constructing the production rules to generate a particular language, targeted language. So, in this connection let me just demonstrate construction of a grammar for this language.

(Refer Slide Time: 13:55)



So, if you consider a string a 1, a 2, a n as we are claiming to give a right linear grammar, a production rule I will look for what is of this form or of course, A goes to x. Now, here x is a terminal string in particular I will restrict myself to a particular symbol of the alphabet. So, that way in a derivation for example, a typical derivation what it will be, it will generate first a 1, then you will have some non terminal symbol let me call it as a 1 and then, since as I am claiming in each step I will be generating one, one symbol each.

So, may be a 1, a 2, then some non terminal symbol sorry a 2 and so on, a 1, a 2, a n and say A n and this non terminal symbol, if I terminate with epsilon then I will have that a 1, a 2, a n. So, look here the type of production rule I am targeting a terminal string followed by at most from non terminal symbol, because unless a non terminal symbol is there I cannot produce further to continue this derivation.

So, in the first step I am assuming not a terminal string at least one symbol I am assuming to generate, say it is a 1 and the non terminal symbol, because it is in the right side of the production rule, even right hand of the production rule, I will have something that is a 1. And now if I continue like this in each step, I am assuming to produce one one symbol each say a 1, a 2 here and so on, in the n th step, I am having a 1, a 2, a n with A n as a non terminal symbol here.

But, what is the criteria that a string to satisfy to fall in the language that is the number of a is in the string is odd, so that means, as and when I am producing a in the string, I should somehow remember that at least it is not the count. But, at least I should be able to understand that whether that the number of a is is even or odd, because the possibility is either even or odd.

So when it is in the odd in number, then I would give to possibility to terminate the derivation, when the situation is I have produced, so far even number of a's, then I should take care that the derivation should not terminate. So, that should have a possibility to further continue that is the philosophy one may observe at this point of time; and thus at least I should have two non terminal symbols, one is essentially which allows you continue the derivation further and which will not allow you to terminate.

And one non terminal symbol which allows you terminate the derivation, let me call for the purpose O and E, these are two non terminal symbols, O is for the one at a particular situation, I will have this non terminal symbol O in the derivation. If so far I have

produced odd number of a's, likewise I will use E, the non terminal symbol E if in the derivation is if so far I have produced even number of a's, so thus I am choosing these two.

And you understand that in the beginning of the derivation of course, I do not have anything generated that means, in the beginning of the derivation that is start symbol of course, I am writing we are uniformly following as. So, whatever it is here that is even number of a's, the number of a's is 0, so that means the start symbol that means, the derivation if I start it E, then there after I will continue. So, I will start with E and for example, if I have generated B, for example if I have generated B in the beginning, then the situation is so far I have produced.

Even number of a's and thus I can connect it to E here, because as I have mentioned at a particular ((Refer Time: 18:28)) point if I am getting E, that means so far I have produced even number of a's. Here there are 0 number of a's and thus I can have this kind of step, suppose if I have produced in the beginning a, then since I have, so far odd number of a's in the derivation we will have O here.

So, let me look at the first step this way, now this situation is if I have odd number of a's, I should have the possibility to terminate the derivation, whereas in case of E I should not have. That means, E will not be terminated, that means we do not introduce the production rule E goes to epsilon sort of, but can introduce the production rule this O goes to epsilon. And now clearly you can see that I have a production rule E goes to b E, because I am saying this is one step, similarly here E goes to a O is one production rule.

In this context suppose if I further continue and suppose, if you are produced a, then as this production rule indicates that you have produced odd number of a's I can have this kind of thing, E goes to a O that production rule that I can use. So, the production rule so far, let me write here E goes to b E, E goes to a O this two production rules that I have. And for example, now when you have non terminal symbol O and if you have produced one more a, what is the possibility now you have two a's that means, even number of a's.

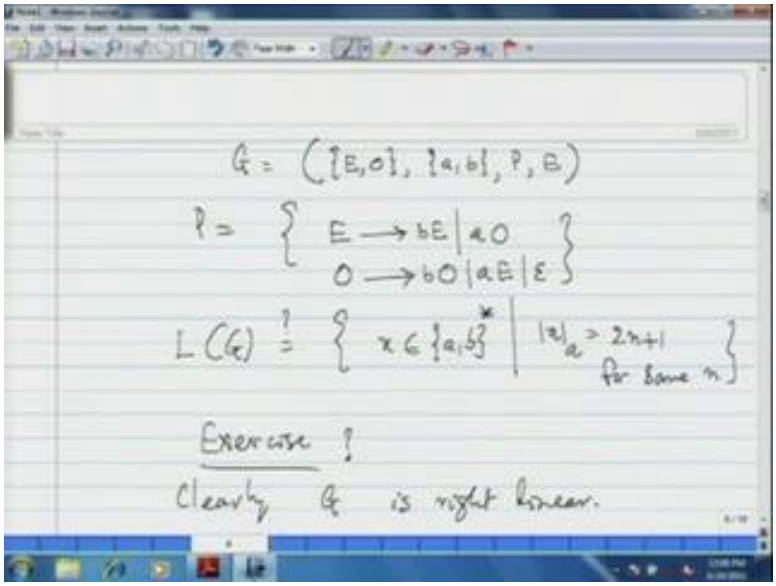
And this should be indicated by giving the non terminal symbol e in the derivation, that means b a a E for example, if I have produced one more a that means, now this O when you are with the non terminal symbol O in sentential form, if I produce a, then I will connect this two E. And if I produce b, then the number of a's so far generated is not

going to change and thus, if there are odd number of a's, then it is still odd number of a's, so O goes to b O.

And as I had mentioned we give the possibility to terminate O through this epsilon, and we do not give the possibility E goes to epsilon, because if you have E in the sentential form there the number of a's is even and we should not terminate the derivation. So, with this philosophy these are the production rule that we have discussed, and you can observe that this production rules precisely generate all those strings which are having odd number of a's.

Let me a give a derivation, this is the production rules remember that, now what is the grammar that I have to mention, because I have not used S here, as per our convention if I just use the production rules, then S is the non terminal symbol which is always start symbol, in this context we will consider E to be the start symbol.

(Refer Slide Time: 22:01)

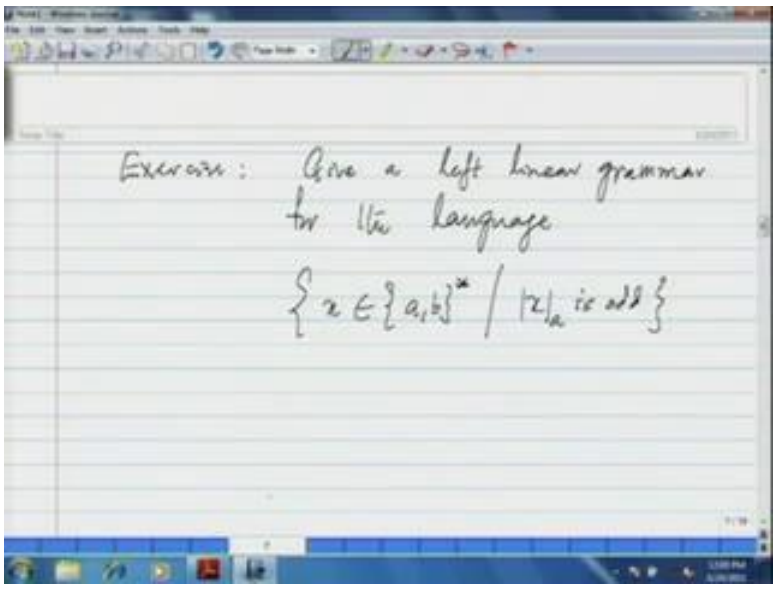


So, the grammar G with E O as non terminal symbols and a, b as terminal symbols and production rules and E as a start symbol is grammar, with the production rules as we have come here, E goes to b E or a O, O goes to b O or a E or epsilon these are the production rules, O goes to b O or a E or O goes to epsilon. So, this production rules if you consider, the language generated by the grammar is as desired that means, all those strings x in a, b star such that, the number of a's in x is odd.

So, you consider typical derivation and observe that, typical derivation which derives a terminal string and observe that string is in this language and for every string in this language you can give a derivation in this grammar, that we can observe. So, take it as an exercise, exercise is to show this thing, the language generated by this grammar is precisely generating those strings which are having odd number of a's; and now you look at here, the production rules that I have obtain here.

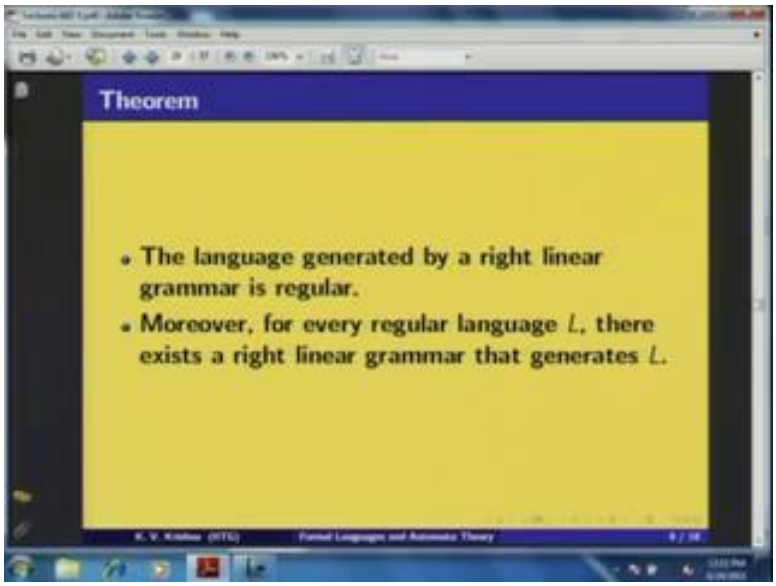
Every production rule has on the right side at most one non terminal symbol and thus this a linear grammar, and more over the non terminal symbol that is appearing on the right side is appearing at the right end and thus you can see clearly this is. So, this question mark is your exercise and clearly this grammar G is right linear, now you can follow this philosophy and somehow to see that, you can give a left linear grammar for this language, so you can take this is an exercise.

(Refer Slide Time: 24:41)



So, another exercise here, give a left linear grammar for the language x in a, b star such that, number of a's in x is odd, so for the same language you can give a left linear grammar. So, we have understood that this language can be generated through a right linear grammar.

(Refer Slide Time: 25:31)



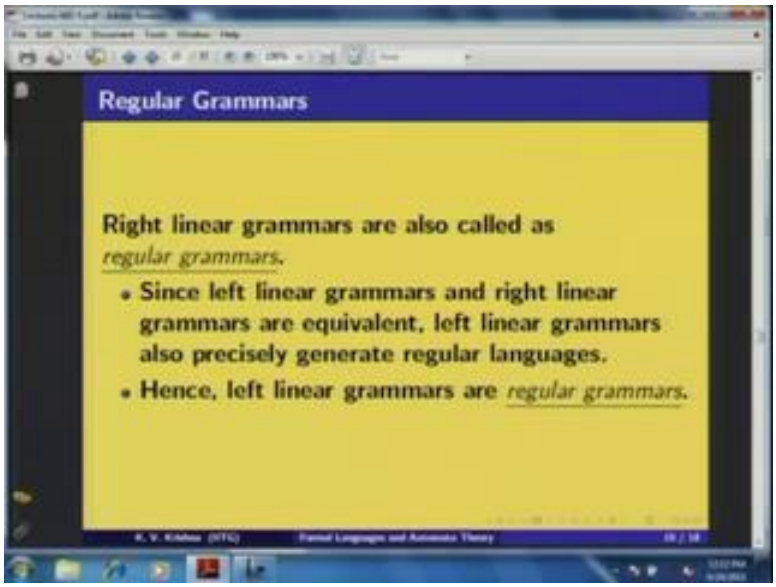
Now, let us look at this theorem, the language generated by a right linear grammar is regular, now if you recollect because what are the examples just before we have discussed, ((Refer Time: 25:48)) this language when we have introduced regular languages to regular expressions. If we remember this language is introduced and mention that giving a regular expression for this language is little bit difficult and we are going to give certain tools to represent this and understand that, it is a regular language.

So, in this context the concept of grammar, particularly the concept of right linear grammar is given here and observed that, this language can be generated through a right linear grammar. And as we have targeted saying that grammars will be a better tool and this theorem is essentially capturing this philosophy, mentioning that the language generated by right linear grammar is regular. That means, from this theorem you can conclude that, the language just mentioned is in fact, a regular language.

Of course, we have not observed through a regular expression, but that philosophy is captured through this theorem, in fact any regular language can be generated through a right linear grammar, that is what is here. Now, proof of this theorem is little bit involved to other we can prove it here, but it will little bit intricate and this point of time, and we will be introducing better tools to prove this theorem and you can have a very elegant proof later, so we will postpone the proof of this theorem.

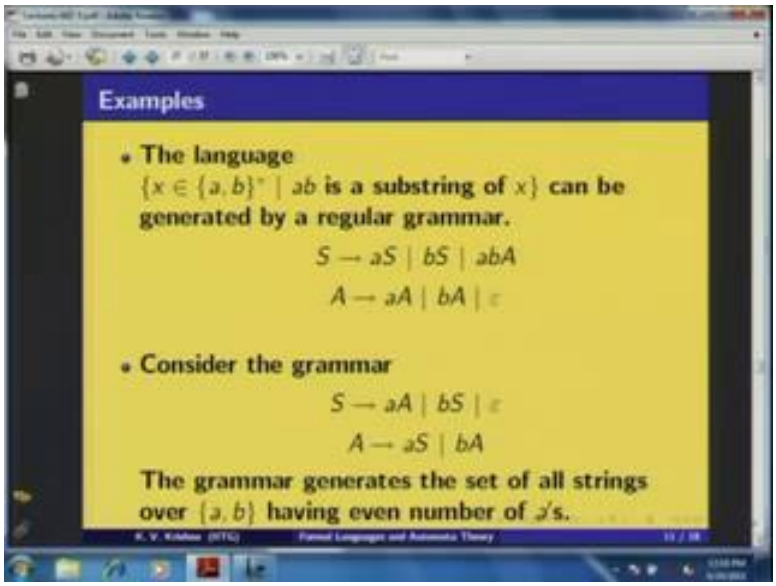
And of course, what is mention a for every regular language L, there exists a right linear grammar that generates L, so that means what essentially the theorem says that, the regular grammars precisely generates regular languages.

(Refer Slide Time: 27:53)



Now, because this right linear grammars are generating regular languages, we can now call them as regular grammars, and because of the equivalence between right linear grammar and left linear grammars, we can also say that left linear grammars are regular grammars. So, regular grammars here after if we use the term regular grammar that is you can, say it is a right linear grammar or a left linear grammar.

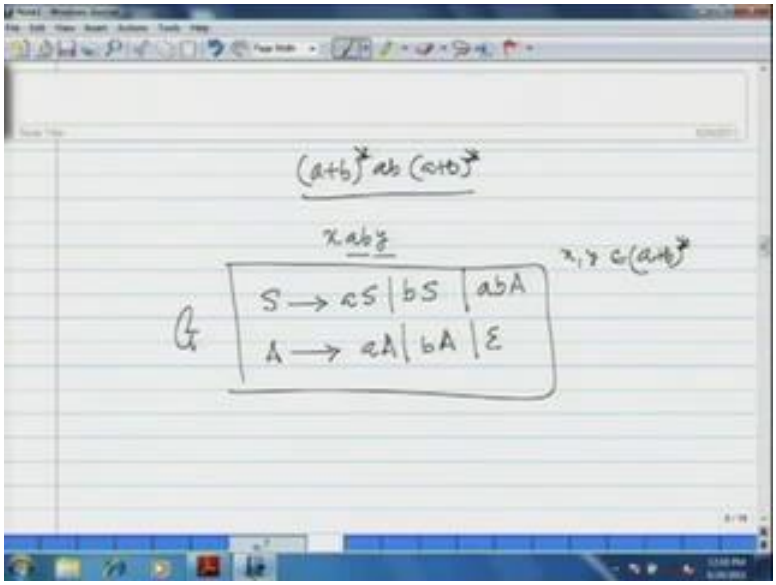
(Refer Slide Time: 28:23)



Now, let us look at some examples that we have already observed that the regular and corresponding rightly in the grammars we will give. So, that you will get familiarity with this right linear grammars and thus what are the languages for which you could not identify regular expressions, you may try through right linear grammar to show that they are regular languages.

This we have already observed that it is regular, because you gave a regular expression for this, because all those strings in which a, b as a sub string this is a regular language.

(Refer Slide Time: 29:00)



What is the regular expression for that, that is a plus b whole star a b a plus b whole star, so this is the regular expression that represents this language, and that this is a regular language. Now, if you look for a right linear grammar for this, you will get the familiarity how to give a right linear grammar for a regular language and then, for regular languages, if you analyze this strings and if you give corresponding right linear grammars, because of that theorem you can conclude that they are regular languages.

Although you do not have regular expressions, you may not be able to regular expressions at this point of time, once we prove that theorem by that time you will be able to give you an regular expression also, although it may be very cumbersome to see and tedious to write. You may certainly give regular expression for the regular languages, so what is the right linear grammar or a regular grammar, for this ((Refer Time: 29:59)).

If you consider these production rules, because again the philosophy is the non terminal symbols should represent in a derivation what is so far generated, if you have a particular non terminal symbol in a particular sentential form in a derivation, it should give you the opinion, you should give the idea that what is the string, so far that you have generated. So, for that purpose here we have considered S and A and of course, once I say this is the grammar, then S is a start symbol of that grammar.

Now, this production rules, because if you look at any string in this, it is some x a b y, for x y with any number of a's, I mean with any combination of any a's and b's, that is an element of a plus b whole star, but you should have at least one a, b that is what is the typical string. So, the production rules here given are essentially to produce any string with the combination of a's and b's, that production rules should be of this form a S or b S, because this production rules will generate any string with a's and b's in any combination.

But, once you produce a, b, then again you can generate any string, so that is the situation here, but you should not terminate S unless you produce a, b, because once you have a freedom here you might be just producing b's, but you have not produced any a, b. Or you would have produce several a's, but you have not produced a, b following to that, so that way we should not terminate the symbol non terminal symbol S, so I do not have I should not introduce the production rule S goes to epsilon here.

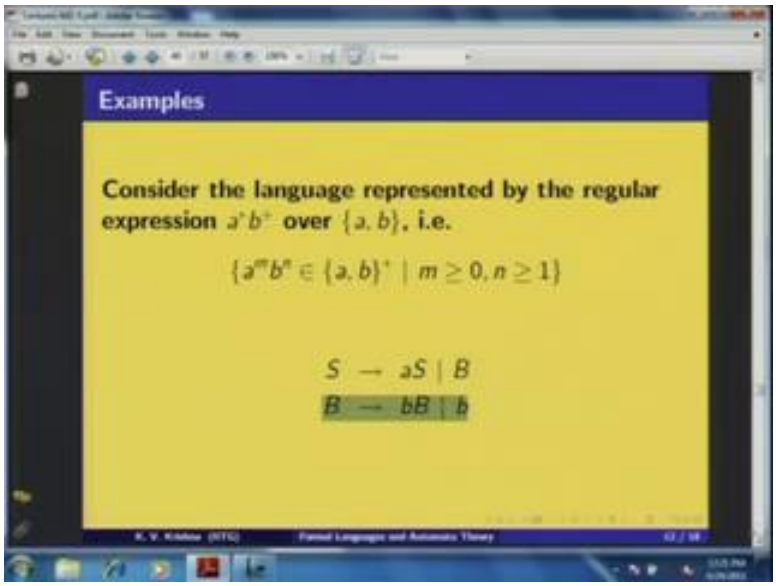
Rather we connect to another non terminal symbol, which will be used to terminate the derivation, but that non terminal symbol I will introduce only when we have produced a, b in a derivation. So, for that purpose if you have produced a, b, then I connect to the non terminal symbol a, this non terminal symbol a now can produce any string over a, b that means, again this rules and this you can terminate whenever you want.

So, thus these are the production rules, if you consider we can observe that the grammar with these production rules precisely generates this language, again here you take it as an exercise to prove that the grammar G generates the language has desired. And you understand that this is a right linear grammar again, because any production rule here we observe, if a non terminal symbol is appearing on it is right hand side, that is appearing at the right end of the production rule here ((Refer Time: 33:26)) or here or anywhere you see.

Wherever the non terminal symbol is appearing on it is right hand side, you have that is appearing at the right end. Now, let us consider ((Refer Time: 33:40)) this grammar, S goes to little a capital A little b capital S or epsilon and this A goes to little a capital S or little b capital A consider this and you can quickly see that this is a right linear grammar. And now you can think of what is the further right linear grammar, what is the language associated to this grammar you can think of that, and observe that particular language, because this is a right linear grammar you can understand that it is also a regular language.

You can think about this as an exercise and observe that, the corresponding language for this grammar is regular, it is easy to understand we have already discussed one similar such example. If you analyze the strings generated by this you can quickly ascertain what is the general representation of a string in the language and thus, you can give the language in set builder form to understand that language is regular, because it is generated through a right linear grammar.

(Refer Slide Time: 35:01)

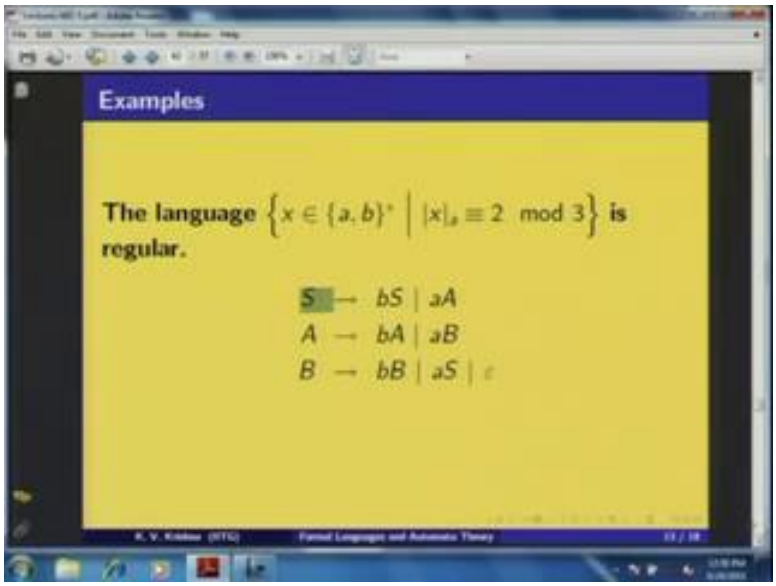


Now, let us consider this regular expression a star b plus and thus, you understand that, that is the regular language the language represented by this is regular language, what is that this is although strings of the form a power m b power n such that, m greater than or equal to 0, n greater than equal to 1. Since it is a star here, you can have epsilon also here, 0 number of a's, so that means strings of the form a power m b power n m can be 0, but b this n, the number of b is here the following b is cannot be non zero.

Here the number of b is should greater than or equal to 1, because b plus is used here, now you can give a right linear grammar for this, again here, because you require at least one b. So, unless you produce b you should not have the termination of the derivation and of course, you need not produce S, that is why the production rule S goes to B, if you use directly S goes to b in the first step, you are not going to produce any S.

And after that you are going to produce any number of b's at least one b of course, and any number of b's you can produce and terminate the derivation by producing b's. If you want that leading is of course, you have to use this production rule as many S, as you want and then you connect to b and then, produce number of b's as required and then, terminate the derivation. And observe that this grammar with of course, S as the start symbol, this grammar produces this language, and understand that this right linear grammar is generating this regular language.

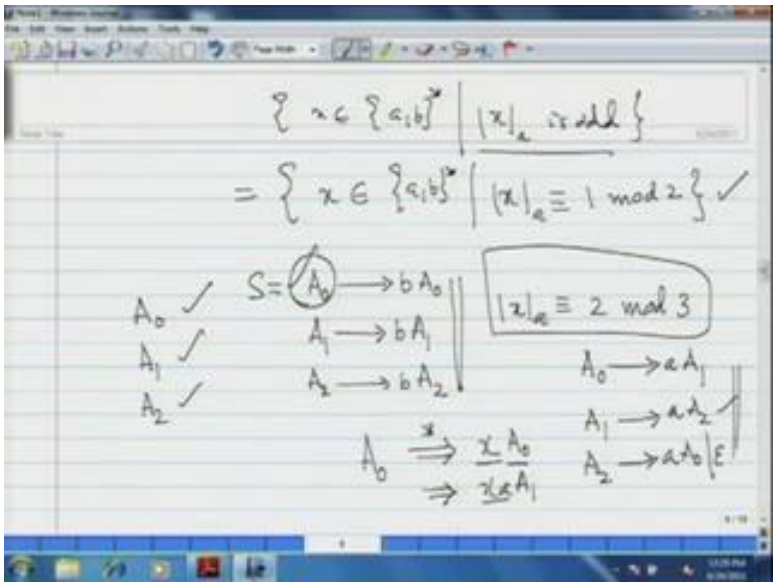
(Refer Slide Time: 37:06)



Now, little more complicated example, that you would not have attempted to give a regular expression for this as well, the number of a is in a string or a, b is 2 mod 3. That means, if you take a string and count the a's divided by 3, the remainder should be 2, that is what the number of a's is congruent to 2 mod 3 means this. So, let us look at giving, because we may not be able to think about a regular expression for this, even for the earlier language that is number of a's is congruent to 1 mod 2.

That means, that is odd number of a's, for that itself we have postponed to discuss it is regular, it is similar to that, but little more even complicated. But, once you take the philosophy of the pervious example you can give a right linear grammar for this; and hence, you can observe this languages regular.

(Refer Slide Time: 38:24)



What is pervious example first let me write the previous example, that is number of a's x is odd, so this can be written as set of all x in a, b star, so I am trying to write in this form, number of a's in x is congruent to 1 mod 2. So, that means, when you divide the number of a's by 2 the remainder is 1, that is when you call the number of a's is odd, now what is the present example and we know a right linear grammar for this language, we have already discussed right linear grammar for this.

Now, in the present context what we are considering, the number of a's in x is congruent to 2 mod 3, so in the previous example we have considered two non terminal symbols. Because, when you divide the number of a's the possibility of remainder either it is 0 or 1 that means, either even number of a's you would have or odd number of a's you would have, and that way we have consider two non terminal symbols.

And when you are having in a sentential form O, which is representing odd number of a's, then you gave the possibility of terminating it, otherwise you are not giving that means, you have to continue to produce. Similarly, here since you are dividing with 3 you can now consider and you look at the number of remainders possible that is 0, 1, 2. So, among these for these three remainders you give three non terminal symbols to start with maybe you can call them as a 1, a 2, a 3.

Or a 0, a 1, a 2 to represent, that is representing the remainder a naught, may be you call it as a 1, a 2, these are non terminal symbols a naught... If it is appearing in the sentential

form; that means, I have produced the number of a's congruent to 0 mod 3 that means, the remainder is 0. And when you have A 1, that means the number of a's is congruent to 1 mod 3 that means, the remainder when you divided by 3 is 1 and similarly, A 2 the remainder is 2.

Now, in the beginning when I am starting the number of a's of course, so the non terminal symbol A 1 should be the start symbol and if I am producing any b's I am not going to worry to count there. So, that way A 1 goes to say b A 1, similarly this that this A 1 is not start symbol, because the number of a's in the beginning is A naught, when the number of a's will be 0 and thus the remainder will be 0, so A naught will be declared as the start symbol.

And the number of b's that you are producing in A naught we are not going to count and that way, this production rule will help you to produce as many b's as you want, when you have A 1. And similarly when you have A 1 you can produce as many b's as you want similarly for A 2, so these are the production rules, which are not worried to count the number of b's that you want to produce in a string.

And now in a sentential form, say we have started with this start symbol A naught and after finitely many steps, you have produced some string x and suppose you have A naught here, as we are targeting right linear grammar, the non terminal symbol will always be at the right end. So, that is how I am writing this way, suppose you have A naught here, what is the meaning of this the number of a's in x is congruent to 0 mod 3.

That means, so far when you divide that you have zero number of I mean the remainder will be 0 on the number of a's, that is the meaning if you have A naught, now if you have produced one more A. Because, as far as b is concerned we have already discussed about the production rules here, these three production rules are given, suppose if you want to produce a, then what it should become this A naught, so since you are adding one more a the remainder will become now 1.

So, thus if you are from A naught, if you are producing one more a, then I have to connect it to A 1 and similarly, if I have A 1 in this place, the number of a's that are produced, say let me now use this production rule and the continue this derivation that is x a A 1. So, that means, in x a I have the number of a's in x a is congruent to 1 mod 3, if

you have one more a generated, that means from A 1, if you want to generate one more a you have to connect it to A 2, because A 2 is representing that remainder two.

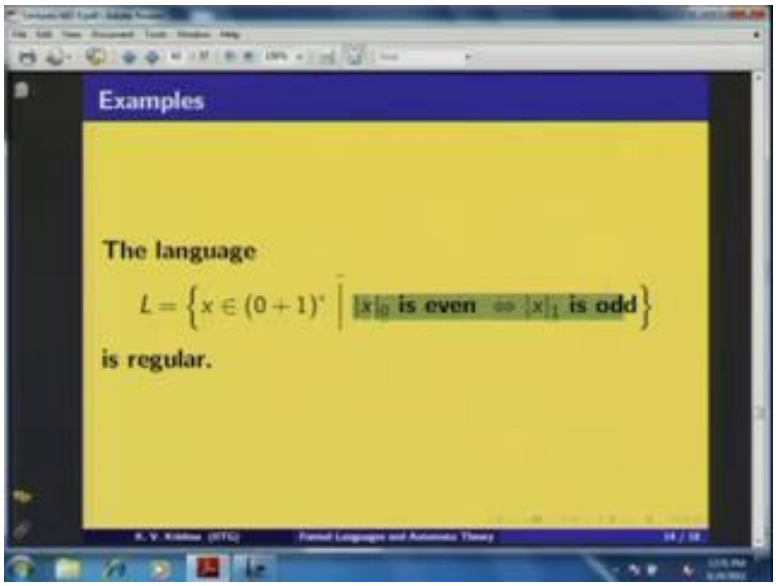
So, this production will can be introduce that way and for example, when you are at A 2, if you have produced one more a, then the remainder will become 0, so it should be connected to A naught. So, along with these three rules, if we include these three rules and where we have to terminate, because our target is to produce all those strings in which the number of a's is congruent to 2 mod 3, whenever in a string if you have produced with this condition, that means you have the non terminal symbol A 2 in that.

So, this non terminal symbol A 2 can be terminated, so the production rule or epsilon, I mean A 2 goes to epsilon can be introduced for this purpose and thus, you understand systematically that this 6 plus 6 plus 1. So, total seven rules what is the derivation you consider, you can produce any string in which the number of a's is congruent 2 mod 3 and conversely. You consider any string in which you have a's and b's and number of a's is congruent to 2 mod 3, if you consider then you can produce using you can produce that string, you can derive that string in this grammar, with this seven rules.

So, the grammar with this seven rules with start symbol A naught that is important here, with start symbol A naught can generate this language, the desired language. Of course, here I have mentioned with S ((Refer Time: 45:41)), that is A naught and a is taking the role of A 1 and b is taking the role A 2. Because, a is when I have in a sentential form when s is appearing the start symbol, that means the number of a's is congruent to 0 mod 3 that is the start symbol.

So, S is equal to A naught A is equal to A 1 B is equal to A 2 you consider and this seven rules generates this language and hence, this language is regular, because this is a right linear grammar and hence, the language is regular.

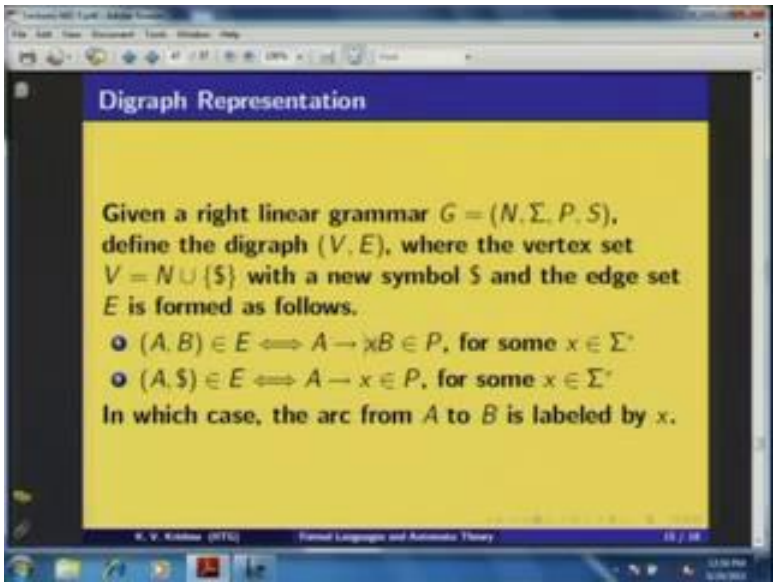
(Refer Slide Time: 46:20)



Now, let me give one more example that number of zeros in string is even, if and only if number of one is in a string is odd. So, if you consider all those strings in over 0, 1 with this condition, number of zeros in x is even if and only if, number of one is in x is odd. The condition itself is looking little bit complicated you can of course, what you can do you can little bit analyze this condition, this logical statement you can little bit analyze, and give the appropriate conditions sort of primitive conditions.

And you may try for a right linear grammar also called regular grammar for this to understand this is regular of course, trying for a regular expression is any way difficult, and a simple point for you is you may trying for a regular grammar for this. But, it will also be tedious when to analyze and understand, we will introduce even better tools to understand regular grammars further and you can understand that this is regular in future. So, in order to go in the direction of that, in order to introduce a better tool to understand regular grammars.

(Refer Slide Time: 47:54)

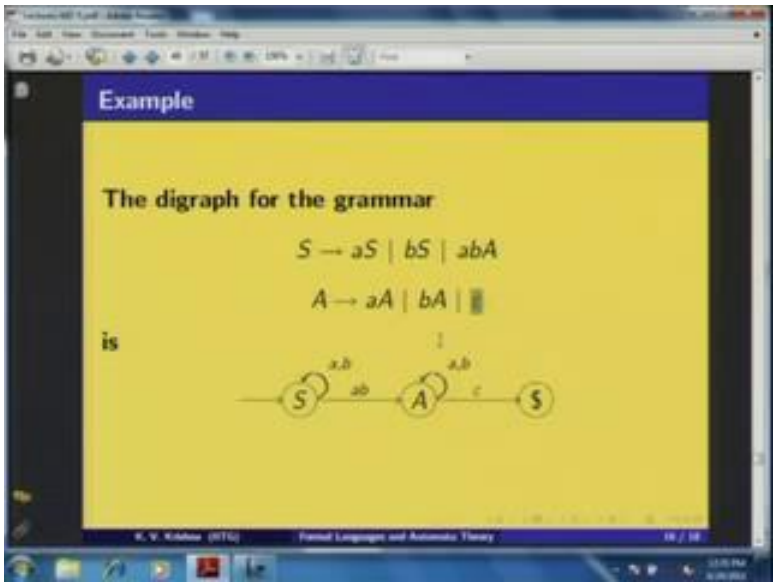


We first give one representation that is called digraph representation for this regular grammar or right linear grammars, and this representation will give you the facility to understand a better tool for regular languages that is the digraph representation. So, this digraph representation, I will introduce this digraph representation for a regular grammars. So, given a right linear grammar G N sigma P S, we define a digraph V E, where the vertex set is the non terminal symbols union with a new symbol, I consider say dollar, and the edge set E is formed as follows.

If you take a production rule that is of the form a terminal A goes to x B form or A goes to x, so when you have the production rule A goes to x B, then I make the non terminal symbols a, b their vertices in the digraph. I will make them adjacent, you will put arc from a to b or if you have the production rule of the form A goes to x, then I consider the edge between from a to the new symbol dollar of course, what are the arcs that I am introducing here we will label them.

So, this is a label digraph as matter of fact, that is labeled by whatever is the whatever is the string that you have here that x will be the label of that arc, so that is how we are defining this digraph representation for a right linear grammar.

(Refer Slide Time: 49:31)

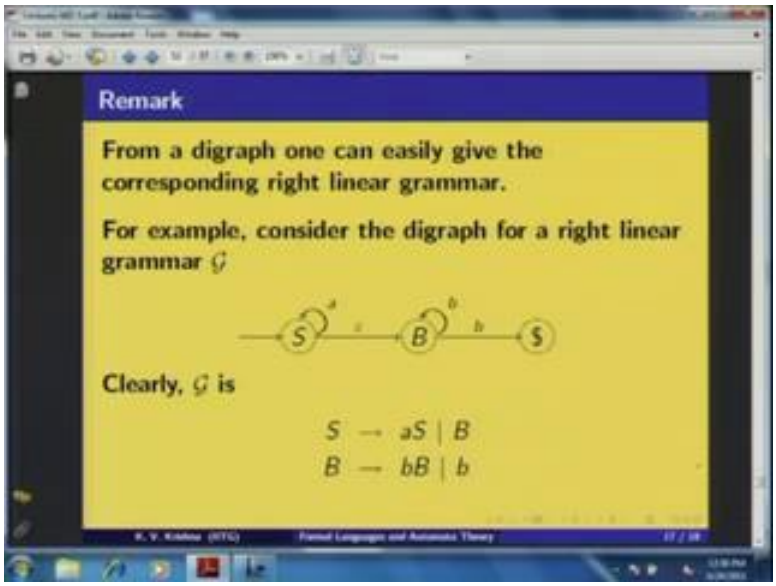


Now, let us look at an example, if you consider this right linear grammar this is the digraph corresponding to that, because you see S A there are the two non terminal symbols. I have two nodes here S and A and a new note with dollar and whatever is the production rule that you consider, accordingly the arcs are introduced, where S goes to a S or b S that means, you have an arc from S to S labeled a of course, we have another arc that means, a essentially loop here from S to S.

Labeled b also we are simply putting one loop here and labeling a comma b, similarly when you connect from S to A, you have the terminal string a, b, so that is the label of this arc going from S to A. And in A you have a loop for a and b that is how it is given and now when you have a terminal string, that is only thing here that S goes to epsilon is only such production rule.

So, those production rules will give you arcs from that particular non terminal symbols to the new symbol dollar, that is what is here, this epsilon is connecting from A to dollar. So, thus this representation this example I hope illustrates you, how to give a digraph for a right linear grammar; not only that if you are given a digraph representation for right linear grammar you can write the corresponding production rules, that is not a big deal.

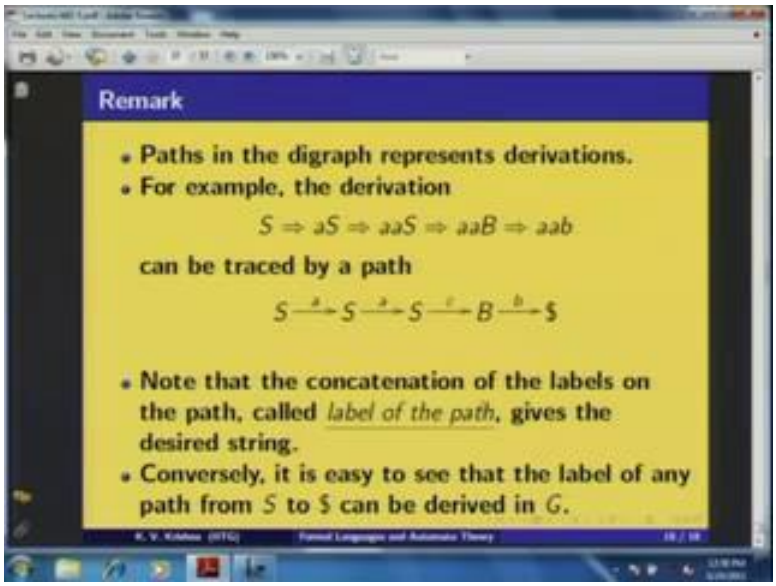
(Refer Slide Time: 50:56)



Look at for example, this digraph for a right linear grammar, you can quickly say what are the corresponding production rules, you can say that these two are non terminal symbols in that S and b are the non terminal symbols. And the production rules are S goes to a S is the production rule and you can say that, S goes to b is a production rule, because epsilon, epsilon b means, S goes to b is production rule. And you have a production rule b goes to b B, b capital B and you are terminating with the production rule B goes to little b, because that is going to dollar.

So, these are the production rules for that, so given a digraph representation for a right linear grammar, you can write the corresponding production rules also.

(Refer Slide Time: 51:48)



Now, here one important philosophical point I will introduce, which gives you the better understanding when we are talking about a better tool for regular languages. The pass in the digraph represent derivations, particularly if when we are interested from the start symbol to, the pass from start symbol to the dollar. Now, if you consider the derivation S goes to a S, that goes to a a S and goes to a a B and then. that goes a a little b, correspondent to this path the previous example, you can quickly see that this is the corresponding path.

And the concatenation of the labels on this path, called labels of the path here after we may call that gives a desired string, because if you concatenate a a epsilon b, if you concatenate this labels, that is the label of this path that you can see this is string that you are deriving from the start symbol S in a derivation. So, that means, if you take the labels of the path from the start symbol S to dollar that give a corresponding to which you can of course, have a derivation in the grammar, and that string can be generated to the grammar.

So, through this digraph representation, we can understand that the labels of the pass from start symbol to the dollar, which are the corresponding derivation will give you the derivation for the string and conversely. So, we can easily see the correspondence between these two derivations and paths the correspondence and you can see that, this

concept will give you better understanding about derivations, through this digraph representation.